UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 10/712,473 | 11/12/2003 | Robert E. Ober | J0658.0014 | 9335 |

38881 7590 07/05/2007

DICKSTEIN SHAPIRO LLP
1177 AVENUE OF THE AMERICAS 6TH AVENUE
NEW YORK, NY 10036-2714

| EXAMINER | |
|---|---|
| JOHNSON, BRIAN P | |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2183 | |

| MAIL DATE | DELIVERY MODE |
|---|---|
| 07/05/2007 | PAPER |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

**BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

Application Number: 10/712,473
Filing Date: November 12, 2003
Appellant(s): OBER ET AL.



Laura C. Brutman
Registration No. 38,395
For Appellant

**EXAMINER'S ANSWER**

This is in response to the appeal brief filed 26 February 2007 appealing from the Office action mailed 25 July 2006.

**(1) Real Party in Interest**

A statement identifying by name the real party in interest is contained in the brief.

**(2) Related Appeals and Interferences**

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected or have a bearing on the Board's decision in the pending appeal.

**(3) Status of Claims**

The statement of the status of claims contained in the brief is correct.

**(4) Status of Amendments After Final**

No amendment after final has been filed.

**(5) Summary of Claimed Subject Matter**

The summary of claimed subject matter contained in the brief is correct.

**(6) Grounds of Rejection to be Reviewed on Appeal**

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

**(7) Claim Appendix**

The copy of the appealed claims contained in the Appendix to the brief is correct.

**(8) Evidence Relied Upon**

Hobbs, U.S. Patent No. 5,197,138

Radhakrishna, U.S. Patent No. 6,823,414

**(9) Grounds of Rejection**

The following ground(s) of rejection are applicable to the appealed claims:

***Claim Rejections - 35 USC § 102***

1. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

2. Claims 1-5, 9-16, 20-22, 24-26, 29-32 and 34-35 are rejected under 35 U.S.C. 102(b) as being anticipated by Hobbs (U.S. Patent No. 5,197,138).

3. Regarding claim 1, Hobbs discloses a method for operating a multi-threaded system having a plurality of active threads (col 1 lines 63-66), the method comprising: assigning an interrupt priority value to each of a plurality of interrupts (col 2 lines 46-50); specifying an interrupt threshold value; and processing a requested interrupt only when the interrupt priority value of the requested interrupt is higher than the interrupt threshold value (col 2 lines 54-57).

4. Regarding claim 2, Hobbs discloses the method of claim 1, wherein processing the requested interrupt comprises: performing an interrupt entry process to prepare for

an interrupt service routine (ISR); executing the ISR (col 2 lines 6-13); and performing an interrupt exit process to return control from the ISR (col 2 lines 14-19).

5. Regarding claim 3, Hobbs discloses the method of claim 2, wherein performing the interrupt entry process comprises: identifying one of the plurality of active threads as an interrupt thread; switching to the interrupt thread if the interrupt thread is not executing (col 2 line 67 to col 3 line 3); and branching to the ISR (col 2 lines 6-13).

6. Regarding claim 4, Hobbs discloses the method of claim 3, wherein each of the plurality of active threads comprises a thread context, and wherein performing the interrupt entry process further comprises saving the thread context of the interrupt thread (col 1 lines 57-64).

7. Regarding claim 5, Hobbs discloses the method of claim 4, wherein performing the interrupt exit process comprises: executing a return from exception (RFE) instruction (col 2 lines 14-19); and restoring the thread context of the interrupt thread (col 1 lines 60-63).

8. Regarding claim 9, Hobbs discloses the method of claim 3, wherein each of the plurality of active threads consists of a first set of context registers (col 1 lines 60-63) and a second set of context registers (col 1 lines 60-63),

*Note that the saving of the context registers is unique for each thread. The use of multiple threads (col 1 lines 63-66) suggests a second set of context registsers.*

Wherein performing the interrupt entry process further comprises saving the first set of context registers of the interrupt thread (col 1 lines 60-63), and wherein executing the ISR comprises: saving the second set of context registers of the interrupt thread if the second set of context registers of the interrupt thread are required for servicing the requested interrupt (col 1 lines 60-63); servicing the requested interrupt (col 2 lines 12-14); and restoring the second set of context registers of the interrupt thread after servicing the requested interrupt if the second set of context registers of the interrupt thread were required for servicing the requested interrupt (col 2 lines 14-19).

9. Regarding claim 10, Hobbs discloses the method of claim 9, wherein performing the interrupt exit process comprises: executing a return from exception (RFE) instruction (col 2 lines 6-13); and restoring the upper context registers of the interrupt thread (col 2 lines 14-19).

10. Regarding claim 11, Hobbs discloses the method of claim 1, further comprising processing traps only in the active threads originating the traps (col 2 lines 22-32).

*Note that all the instances provided refer to responding to traps in the active threads that originate the trap.*

11. Regarding claim 12, Hobbs discloses the method of claim 11, wherein processing traps comprises: detecting a trap from an originating thread, the originating thread being one of the plurality of active threads (col 3 lines 7-10); storing trap background data for the trap if the trap is asynchronous (col 3 lines 24-29); and associating a trap pending indicator with the originating thread if the originating thread is not executing (col 3 lines 40-50).

*Note in line 44 the use of the term "delay". This suggests that there is an indicator waiting for a change to the originating thread.*

12. Regarding claim 13, Hobbs discloses the method of claim 12, wherein processing traps further comprises: performing a trap entry process to prepare for a trap handling routine (col 3 lines 40-50); executing the trap handling routine (col 2 lines 12-14); and performing a trap exit process to return control from the trap handling routine (col 2 lines 14-19).

*Note that, in col 3 line 48, it is stated that an I/O interrupt routine is used for the trap, indicating that a trap uses the same methods for handling as typical interrupts.*

13. Regarding claim 14, Hobbs discloses the method of claim 13, wherein each of the plurality of active threads comprises a thread context (col 1 lines 63-66), and wherein performing the trap entry process comprises: waiting for originating thread to start executing if the originating thread is not executing (col 2 line 67 to col 3 line 3);

saving the thread context of the originating thread (col 2 lines 6-13); and branching to a trap handler (col 2 lines 12-14).

14. Regarding claim 15, Hobbs discloses the method of claim 14, wherein waiting for the originating thread to start executing comprises monitoring the plurality of active threads until execution switches to one of the plurality of active threads associated with the trap pending indicator (col 3 lines 40-50).

15. Regarding claim 16, Hobbs discloses the method of claim 14, wherein performing the trap exit process comprises: executing a return from trap instruction; and restoring the context of the originating thread (col 2 lines 14-19).

*Note, again, that a trap is handled as an interrupt.*

16. Regarding claim 20, Hobbs discloses the method of claim 13, wherein each of the plurality of active threads consists of a first set of context registers (col 1 lines 60-63) and a second set of context registers (col 1 lines 60-63),

*Note that the saving of the context registers is unique for each thread. The use of multiple threads (col 1 lines 63-66) suggests a second set of context registsers.*

Wherein performing the trap entry process further comprises saving the first set of context registers of the originating thread (col 2 lines 7-12), and wherein executing the trap handling routine comprises: saving the second set of context registers of the

originating thread if the second set of context registers of the originating thread are required for servicing the trap (col 3 lines 40-50);

*Note that, as explained in the citation, a context switch may be necessary which, as indicated in col 1 lines 57-63, requires saving the context into memory.*

Servicing the trap (col 2 lines 12-14); and restoring the second set of context registers of the originating thread after servicing the trap if the second set of context registers of the interrupt thread were required for servicing the trap (col 2 lines 15-19).

17. Regarding claim 21, Hobbs discloses the method of claim 20, wherein performing the interrupt exit process comprises: executing a return from trap instruction; and restoring the upper context registers of the originating thread (col 2 lines 15-19).

18. Regarding claim 22, Hobbs discloses a method for operating a multi-threaded system having a plurality of active threads (col 1 lines 63-66), the method comprising: accepting a request for an interrupt; switching execution to a predetermined one of the plurality of active threads (col 2 line 67 to col 3 line 3); and executing an interrupt service request from the predetermined one of the plurality of active threads to service the interrupt (col 2 line 12-14).

19. Also regarding claim 22, Hobbs discloses the method of claim 22, wherein accepting a request for an interrupt comprises: assigning a unique interrupt priority value to each interrupt for the multi-threaded embedded system (col 2 lines 47-50);

specifying a common threshold interrupt value for all the active threads; and taking the interrupt only when the unique interrupt priority value of the interrupt is higher than the common threshold interrupt value (col 2 lines 54-57).

20. Regarding claim 24, Hobbs discloses a method for operating a multi-threaded embedded system having a plurality of active threads (col 1 lines 63-66), the method comprising processing traps only in the active threads originating the traps (col 2 lines 22-32).

21. Regarding claim 25, Hobbs discloses a multi-threaded system comprising: thread execution logic for generating instruction requests from an executing thread (col 2 lines 3-6); and threshold interrupt logic for generating an interrupt threshold value, wherein the thread execution logic only accepts interrupts having an interrupt priority higher than the interrupt threshold value (col 2 lines 54-57).

22. Regarding claim 26, Hobbs discloses the multi-threaded system of claim 25, further comprising interrupt thread logic for switching execution to a selected interrupt thread before servicing any interrupt (col 2 lines 67 to col 3 line 3).

23. Regarding claim 29, Hobbs discloses a multi-threaded system comprising: means for specifying an interrupt threshold value; and means for processing a

requested interrupt only when an interrupt priority value of the requested interrupt is higher than the interrupt threshold value (col 2 lines 54-57).

24. Regarding claim 30, Hobbs discloses the multi-threaded system of claim 29, wherein the means for processing the requested interrupt comprises: means for identifying one of the plurality of active threads as an interrupt thread (col 2 lines 3-6); means for switching to the interrupt thread if the interrupt thread is not executing (col 2 line 67 to col 3 line 3); and means for branching to the ISR (col 2 lines 12-15).

25. Regarding claim 31, Hobbs discloses the multi-threaded system of claim 30, wherein the means for processing the requested interrupt further comprises means for saving a thread context of the interrupt thread (col 1 lines 57 to 63).

26. Regarding claim 32, Hobbs discloses the multi-threaded system of claim 31, wherein the means for processing the requested interrupt further comprises means for restoring the thread context of the interrupt thread after executing a return from exception (RFE) instruction (col 2 lines 14 to 19).

27. Regarding claim 34, Hobbs discloses the multi-threaded system of claim 29, further comprising means for processing traps (col 3 lines 7-11), the means for processing traps comprising: means for detecting a trap from an originating thread (col 3 lines 40-46); means for storing trap background data for the trap if the trap is

asynchronous (col 3 lines 24-29); means for processing the trap if the originating thread is executing (col 3 lines 40-50); and means for associating a trap pending indicator with the originating thread if the originating thread is not executing (col 3 lines 40-50).

*Note in line 44 the use of the term "delay". This suggests that there is an indicator waiting for a change to the originating thread.*

28. Regarding claim 35, Hobbs discloses the multi-threaded system of claim 34, wherein the means for processing traps further comprises means for detecting execution of an active thread associated with the trap pending indicator (col 3 lines 40-50).

*Note that, clearly, for the AST to throw and interrupt when its originating thread is being executed, that state must be detected.*

### *Claim Rejections - 35 USC § 103*

29. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

30. Claims 6-8, 17-19, 27-28, 33 and 36 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hobbs (U.S. Patent No. 5,197,138) in view of Radhakrishna (U.S. Patent No. 6,823,414).

31. Regarding claim 6, Hobbs discloses the method of claim 5, wherein performing the interrupt entry process further comprises thread switching (col 2 line 67 to col 3 line 3).

Hobbs fails to disclose disabling interrupts.

Radhakrishna (U.S. Patent No. 6,823,414) discloses an ISR that disables interrupts (col 1 lines 66-67).

In the environment of Hobbs, a system with the intention of organizing a large number and type of interrupts by giving them a priority system, one of ordinary skill in the art would be motivated to allow for an interrupt that is already in the ISR to maintain it's priority rather than potentially being "interrupted" by a further interrupt. As shown in Radhakrishna col 1 line 67 to col 2 line 4, interrupts are disabled "so that the currently-interrupting event can be processed in an unbroken fashion (i.e., without further interrupts from the same device that diverts CPU processing attention before the ISR has completed it's function)."

It would have been obvious for one of ordinary skill in the art to allow the computer system of Hobbs to include the ability to disable interrupts while executing the ISR, as in Radhakrishna.

32. Regarding claim 7, Hobbs/Radhakrishan discloses the method of claim 6, wherein performing the interrupt exit process further comprises enabling interrupts (Radhakrishna col 2 lines 2-4)

*Note that the interrupts are disabled so interrupts can not divert the "CPU processing attention before the ISR has completed its function", clearly suggesting that interrupts will be re-enabled after the ISR has completed.*

And thread switching (col 2 lines 8-19).

*Note that the term "writes in memory contents to registers" suggests a contrext switch (or thread switch).*

33. Regarding claim 8, Hobbs/Radhakrishan discloses the method of claim 6, wherein executing the ISR comprises enabling interrupts (Radhakrishna col 2 lines 2-4) and thread switching (col 2 lines 8-19) after a predetermined interval (col 2 lines 12-14).

*Note that the complete ISR program code is considered to be the predetermined interval.*

34. Regarding claim 17, Hobbs/Radhakrishan discloses the method of claim 16, wherein performing the trap entry process further comprises disabling interrupts (103 with R from claim 6) and thread switching (col 2 line 67 to col 3 line 3).

35. Regarding claim 18, Hobbs/Radhakrishan discloses the method of claim 17, wherein performing the trap exit process further comprises enabling interrupts (Radhakrishna col 2 lines 2-4)

*Note that the interrupts are disabled so interrupts can not divert the "CPU processing attention before the ISR has completed its function", clearly suggesting that interrupts will be re-enabled after the ISR has completed.*

And thread switching (col 2 lines 8-19).

*Note that the term "writes in memory contents to registers" suggests a contrext switch (or thread switch).*

36. Regarding claim 19, Hobbs/Radhakrishan discloses the method of claim 17, wherein executing the trap handling routine comprises enabling interrupts (Radhakrishna col 2 lines 2-4) and thread switching (col 2 lines 8-19) after a predetermined interval (col 2 lines 12-14).

*Note that the complete ISR program code is considered to be the predetermined interval.*

37. Regarding claim 27, Hobbs/Radhakrishan discloses the multi-threaded system of claim 26, further comprising disabling logic for disabling interrupts and thread switching while an interrupt is being serviced (103 with R from claim 6).

38. Regarding claim 28, Hobbs/Radhakrishan discloses the multi-threaded system of claim 27, further comprising thread tagging logic for storing trap background data for asynchronous traps (col 3 lines 7-10 and col 3 lines 24-29), wherein every trap is handled in its originating thread (col 3 lines 40-50).

39. Regarding claim 33, Hobbs/Radhakrishan discloses the multi-threaded system of claim 32, wherein the means for processing the requested interrupt further comprises means for disabling interrupts (103 with R from claim 6) and thread switching (col 2 line 67 to col 3 line 3) during execution of the ISR (col 2 lines 12-14).

40. Regarding claim 36, Hobbs/Radhakrishan discloses the multi-threaded system of claim 35, wherein the means for processing the trap comprises means for disabling interrupts (103 with R from claim 6) and thread switching (col 2 line 66 to col 3 line 4 and col 3 lines 40-50).

*Note that, as indicated in col 3 lines 15-20, the trap throws an interrupt, suggesting that it is treated as an interrupt.*

**(10) Response to Arguments**

**A. Claim 1 is properly rejected because Appellant erroneously contends that the claimed interrupt value is required to be "fixed."**

Appellant states:

*"Independent claim 1 recites 'processing a requested interrupt only when the interrupt priority value of the requested interrupt is higher than the interrupt threshold value.' The benefits of the claimed interrupt threshold value are explained in paragraph 27 of the published application, where it discusses that the fixed interrupt threshold value (ITV) provides a global interrupt priority value for all active threads. Since any interrupt accepted by thread execution logic must have a higher interrupt priority value than the global interrupt threshold value, any interrupt being service will always have a higher interrupt priority than any of the active threads, thereby preventing priority inversion.*

*In contrast, Hobbs discloses that '[a]n interrupt will not be recognized or serviced by the processor until the priority of the code thread is lower than the priority of the interrupt.' (See col. 2, lines 54-57.) Thus, Hobbs compares the priorities of two different processor functions (i.e., code thread and interrupt), wherein each of these priorities varies. This is different from determining whether a priority of an interrupt is higher than a fixed threshold value, as claimed. Therefore, independent claim 1, along with the dependent claims 2-5, 9-16, 20, and 21, is patentable."*

Examiner disagrees. Appellant points out that the current application utilizes a fixed threshold value whereas Hobbs discloses a threshold value that is variable. Therefore, Appellant believes that the "fixed threshold value" as claimed in claim 1 should be allowable. In fact, it appears that Appellant's entire argument hinges on the distinction between fixed and variable interrupt values.

Here is the problem: a fixed interrupt value is not claimed. The word "fixed" does not appear anywhere in claim 1 and there is no language, read either in isolation or as a whole, that would even imply that the value is fixed rather than variable. For support for this fixed value, Appellant directs attention to paragraph 27 of the published application. Paragraph 27 is written as follows:

> *"To avoid priority inversion, thread handling logic 300 processes interrupts using a global interrupt handling methodology in which threshold interrupt logic 320 specifies an interrupt threshold value ITV that controls interrupt handling across all active threads. More specifically, an interrupt is only accepted by thread execution logic 310 if a unique interrupt priority value (IPV) assigned to that interrupt is higher than interrupt threshold value ITV. In this manner, interrupt threshold value ITV effectively represents a global interrupt priority value for all active threads T(0)-T(Z-1). Therefore, since any interrupt accepted by thread execution logic 310 must have a higher interrupt priority value than this global interrupt priority value, any interrupt being serviced will always have a higher interrupt priority than any of the active threads, thereby preventing priority inversion."*

Examiner notes that the word "fixed" is not found anywhere within this paragraph (or anywhere in the specification for that matter). Nothing from this language indicates to Examiner that the threshold value is limited to a fixed value. And even if such language existed, it would still be improper for Examiner to read such language into the claims.

**B. Claim 22 is properly rejected because Hobbs teaches a code priority value that is common to all threads in a program.**

Appellant uses slightly different wording for the argument with respect to claim 22; Appellant states that "[Hobbs] is different from taking an interrupt only when a unique interrupt priority value of the interrupt is higher than a specified common threshold interrupt value, as claimed."

Examiner disagrees. The wording of Appellant's argument is quite similar to that of claim 1, but Appellant notes a distinction between Hobbs and the current application based on a "common threshold interrupt value" rather than a "fixed threshold interrupt value." Claim 22 does disclose, "specifying a common threshold interrupt value for all the active threads." This limitation appears to require a teaching in Hobbs that all threads use a common threshold interrupt value.

Hobbs col. 2 lines 50-54 states:

*"These more complex systems usually include registers which permit respective priorities to be assigned to various interrupts and also permit a priority to be assigned to the program or code thread currently being executed by the processor."*

Note in particular that the priority can be assigned to either a program or a code thread. Assigning a priority to a code thread does suggest that it is possible to assign each thread a different priority—thus, not necessarily teaching the common priority for every thread as required by claim 22.

However, this citation also discloses assigning a priority to the entire program. This embodiment teaches assigning a common priority value to every thread in the program and, consequently, discloses the required teachings to satisfy the claim limitation of a "common threshold interrupt value for all the active threads."

**C. Claim 25 is properly rejected for the same reasons as Claim 22 since the disputed limitation is broader than that of claim 22.**

Appellant's arguments with respect to independent claim 25 also appear to mirror the previous arguments nearly verbatim. Appellant states:

*"Thus, Hobbs compares the priorities of two different processor functions (i.e., code thead and interrupt), wherein each of these priorities varies. This is different from a thread execution logic only accepting interrupts having an interrupt priority higher than an interrupt threshold value, as claimed."*

Here, neither Appellant's arguments or claims specify that the interrupt threshold value must be fixed or common. For the reasons discussed above, the limitation is satisfied and the rejection is proper.

**D. Claim 29 is properly rejected for the same reasons as those discussed above.**

Appellant's arguments regarding claim 29 appear to be the same as those discussed above and are similarly unpersuasive.

**(11) Related Proceeding(s) Appendix**

No decision rendered by a court or Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejection should be sustained.

Respectfully submitted,

Brian P. Johnson

07 June 2007

Conferees:

Eddie Chan

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

Lynne Browne
APPEAL PRACTICE SPECIALIST, TQAS
TECHNOLOGY CENTER 2100